See 1473

12

AD A030688

SOLVING GEOMETRIC PROGRAMS USING GRG-RESULTS AND COMPARISONS

BY

M. RATNER, L.S. LASDON and A. JAIN

TECHNICAL REPORT SOL 76-1
MAY 1976

Systems Optimization Laboratory

Department of Operations Research

Stanford University

Stanford California 94305

DDC
RECEIVED
OCT 13 1976
D

SOLVING GEOMETRIC PROGRAMS USING GRG-RESULTS AND COMPARISONS

ACCESSION for
NTIS White Section ☑
DDC Buff Section ☐
UNANNOUNCED ☐
JUSTIFICATION..................
BY..................
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A

by

M. Ratner, L. S. Lasdon and A. Jain

TECHNICAL REPORT SOL 76-1

May 1976

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH

Stanford University
Stanford, California

Research and reproduction of this report were partially supported by the Office of Naval Research under Contracts N00014-75-C-0267, N00014-75-C-0865; U.S. Energy Research and Development Administration Contract E(04-3)-326 PA#18; the National Science Foundation Grant DCR75-04544 at Stanford University; and by the Office of Naval Research under Contract N00014-75-C-0240 and National Science Foundation Grant SOC74-23808 at Case Western Reserve University.

Reproduction in whole or in part is permitted for any purposes of the United States Government. This document has been approved for public release and sale; its distribution is unlimited.

DDC
RECEIVED
OCT 13 1976

SOLVING GEOMETRIC PROGRAMS USING GRG-RESULTS AND COMPARISONS

By

M. Ratner, L. S. Lasdon and A. Jain

Introduction

This paper describes the performance of a generalized reduced gradient (GRG) algorithm in solving geometric programs. The code used, described in [5], is a general purpose nonlinear programming code, and takes no advantage of the structure of geometric programs. First partial derivatives of the objective and all constraint functions are required, and these are computed by simple forward difference approximations. All problem functions are expressed in power form, i.e., each term, t_i, has the form

$$t_i = c_i \prod_j x_j^{a_{ij}} \quad .$$

Problems Solved and Measures of Comparison

The geometric programs solved come from two sources: 8 problems given by Dembo in [2] and the 24 problems of Rijckaert and Martens in [6]. Problem sizes are given in Table 1 below. The problems are good examples of small, dense, highly nonlinear NLP's. The problems with some negative terms are generalized geometric programs with signomial constraints.

TABLE 1

Problem Size

Problem	No. of variables	No. of constraints	No. of positive terms	No. of negative terms	No. of binding constraints at optimality
D1	12	3	31	0	3
D2	5	6	15	8	2
D3	7	14	31	13	5
D4A,B	8	4	14	2	4
D4C	8	5	16	0	5
D5	8	6	14	5	6
D6	13	13	27	12	11
D7	16	19	40	21	--
D8A,B,C	7	4	18	0	A: 2, B: 3, C: 4
R1	4	2	6	0	2
R2	3	1	9	0	1
R3	4	1	12	0	1
R4	11	3	30	0	3
R5	4	3	8	0	3
R6	8	7	12	0	7
R7	8	7	12	0	6
R8	7	7	48	0	2
R9	2	1	4	1	1
R10	3	1	4	2	1
R11	4	2	6	1	2
R12	8	4	13	2	4
R13	8	6	14	5	6
R14	10	6	13	2	6
R15	10	7	12	3	7
R16	10	7	13	3	7
R17	11	9	14	5	9
R18	13	9	18	4	9
R19	8	5	26	2	5
R21	10	7	16	7	7
R22	9	10	36	21	7
R24	10	10	23	13	8

These may have local optima which are not global (such a point was encountered in at least one problem).

Measures of Comparison

In comparing GRG with the code used by Dembo in [2] (one of the better special purpose GP codes) two measures were available--the final objective value obtained and the "standard time" required to achieve that value. Standard time is the execution time for the problem divided by the time to execute a timing program written by Colville [1]. This program inverts a 40 by 40 matrix 10 times. Use of standard time is supposed to compensate for the effects of different computing environments, e.g., machines, compilers, etc. To investigate this we solved 4 problems on the IBM 370/168 at Stanford University using three different FORTRAN compilers: the FORTRAN H compiler (OPT=2), the WATFIV compiler with the CHECK option and the WATFIV compiler with the NOCHECK option. The results appear in Table 2, which gives the times required by GRG to solve four problems (with minimal printed output) divided by the time required to run the timing program. There is great variation in standard times between the three compliers, with widest variation (by factors of from 3 to 10) between WATFIV (CHECK) and the FORTRAN H compilers. Evidently this naive way of compensating for computing environment is inadequate. To compare with the other published results, we chose the WATFIV NOCHECK compiler, partly for convenience, partly because it gave the median times. In all GRG runs there was no printing of intermediate results, but input data and final results were printed. In

TABLE 2

Standard Execution Times on Three FORTRAN Compilers

Problem	WATFIV (CHECK)	WATFIV (NOCHECK)	IBM FORTRAN H (OPT=2)
D4C	0.026	0.052	0.109
D5	0.025	0.049	0.069
R2	0.005	0.012	0.038
R9	0.003	0.007	0.033
Colville Timing Program (IBM 370/168 c.p.u. seconds)	41.80	16.83	3.91

problems with run times less than 1 second, even this printing may consume a large fraction of total time.

Comparison with the Rijckaert and Martens results is difficult, since their starting points were chosen randomly, and were not published. We chose our starting values so that odd-subscripted variables were one-half their optimal value, and even-subscripted variables were three-halves their optimal value. The resulting points are shown in Appendix A.

Computational Results

Table 3 shows the performance of GRG on the Dembo problems on our first attempt. Problem 1A was too badly scaled to attempt solution, and the code failed on Problems 3, 6 and 7. In Problems 3 and 6, GRG terminated prematurely when no decrease in the objective was achieved while attempting to move in the direction of steepest descent, while in Problem 7 the program terminated short of feasibility at a local optimum of the Phase I objective.

Improved results were obtained by using an alternative pivoting strategy in computing the basis inverse. This strategy allowed pivoting on matrix elements smaller than allowed by the previous strategy if the alternative was entering a variable at a bound into the basis. This avoided degenerate bases in some cases, and allowed solution of problem 3 and improved performance on number 5 (see Table 4).

TABLE 3

Computational Results for Dembo GP Problems, Using Specified Constraint Tolerances

Prob. No.	WATFIV* Time	Std.* Time	Dembo Std. Time	Dembo Opt.	Our Opt.	FCN Calls (n_f)†	Grad Calls (n_g)†	Equiv. FCN Calls (n_e)†	Newton‡ Avg.	Reason** for Term.
1A			.2747	4.8905E9	Too Badly Scaled for GRG					
1B	0.94	.055	.2711	3.168213	3.169247	189	15	369	.34	F.C.
2	0.17	.001	.0024	10127.13	10122.44	17	6	47	1.14	K.T.
3	F	F	.0829	1227.18	1453.23	163	15	268	2.22	ALPH=0
4A	0.65	.038	.2806	3.951698	3.951153	141	18	285	1.21	K.T.
4B	0.58	.034	.1324	3.956197	3.951165	132	16	260	1.35	F.C.
4C	0.89	.052	.0213	3.95207	3.95209	168	17	304	1.94	F.C.
5	0.84	.049	.1255	7049.32	7049.24	174	16	302	1.88	F.C.
6	F	F	.3275	97.5910	261.16	304	21	577	2.74	ALPH=0
7	F	F	.2403	174.7888	Could not find feasible point					
8A	4.75	.282	.0954	1809.762	1809.763	1398	72	1902	3.43	F.C.
8B	3.28	.194	.0955	911.8796	911.8801	878	53	1249	2.82	F.C.
8C	7.46	.443	.0792	543.6664	543.6681	2274	105	3009	4.04	F.C.

*F ≡ Failure

**F.C. ≡ Fractional charge in objective less than 10^{-4} for 3 consecutive iterations

K.T. ≡ Kuhn-Tucker point found to within 10^{-4}

ALPH=0 ≡ Premature termination--no function decrease in direction of steepest descent.

‡Average number of Newton iterations per attempt to solve for basic variables.

†n_f ≡ Number of function calls

n_g ≡ Number of gradient calls

n_e ≡ Equivalent function calls = $n_e + n_f + N \cdot n_g$ where N = number of variables

TABLE 4

Computational Results for Dembo GP Problems, Using Smaller Alternate Pivot

Prob. No.	WATFIV Time (ser)	Std. Time	Dembo Std. Time	Dembo Opt.	Our Opt.	FCN Calls	Grad Calls	Equiv. FCN Calls	Newton Avg.	Reason for Term.
3	0.96	.057	.0829	1227.18	1227.19	228	21	375	2.10	F.C.
5			.1255	7049.32	7049.6	165	14	277	2.28	F.C.

Table 5 shows the effects of another modification to GRG. The code uses the BFS variable metric method to minimize the reduced objective. The original strategy was to update the approximation, H, to the inverse hessian used by this method only when the line search terminated in an unconstrained optimum. Otherwise it was reset to the identity, and the search direction became the negative reduced gradient. The new strategy used the BFS update at each iteration, except those at which a basis change occurred. In the 5 problems of Table 5, this new strategy was better in all problems but one, significantly better in 3 problems.

Some Dembo problems (whose feasibility tolerance specified was tighter than 10^{-4}) were re-run using the default feasibility tolerance (10^{-4}) of the GRG code. As shown in Table 6, solutions wer obtained faster than with the specified tolerances (Table 3). This prompted the use of a coarse tolerance to obtain an initial solution, followed by a refinement using the specified tolerances. As shown in Table 7, this strategy yielded a significant decrease in computational effort for Problems 8A, 8B and 8C.

The performance of GRG on the Rijckaert-Martens problems is shown in Table 8. The column "Reported S.T." contains the best standard time reported by Rijckaert and Martens [6] in a comparison of eleven special purpose codes for geometric programming and one general purpose code. GRG was generally slower than the best code and missed the true optimum by one to two percent in Problems 8, 13 and 15. Otherwise, GRG solved all these problems satisfactorily.

TABLE 5

Computational Results for Selected GP Problems, Updating H-Matrix Whenever Possible

Prob. No.	H-Matrix Reset: WATFIV Time	H-Matrix Reset: Our Opt.	H-Matrix Reset: Equiv. FCN Calls	H-Matrix Reset: Newton Avg.	H-Matrix Updated: WATFIV Time	H-Matrix Updated: Our Opt.	H-Matrix Updated: Equiv. FCN Calls	H-Matrix Updated: Newton Avg.
D8A	3.96	1809.428	1526	2.11	2.57	1809.007	973	1.83
D8B	2.28	911.5566	810	1.98	2.61	911.6840	964	2.34
D8C	3.08	543.5831	1120	2.13	2.87	543.5858	1051	2.42
R14	6.11	1.1436	2939	3.15	2.65	1.1436	1217	2.55
R17	4.03	.1406	1445	1.99	2.70	.14228	1038	1.65

The Dembo problems above had a constraint tolerance of 10**-4.

TABLE 6

Computational Results for Dembo GP Problems, Using Constraint Tolerance of 10**-4

	TOL=10**-4				Dembo Tolerance				
Prob. No.	WATFIV Time	Our Opt.	Equiv. FCN Calls	Newton Avg.	Tol.	WATFIV Time	Our Opt.	Equiv. FCN Calls	Newton Avg.
1B	0.61	3.176152	230	0.49	10**-6	0.94	3.169247	369	0.34
3	F	1452.74	219	1.64	10**-5	F	1453.23	268	2.22
6	F	249.18	501	1.53	10**-6	F	261.16	578	2.74
8A	3.96	1809.428	1526	2.11	10**-6	4.75	1809.763	1902	3.43
8B	2.28	911.5566	810	1.98	10**-6	3.28	911.8801	1249	2.82
8C	3.08	543.5831	1120	2.13	10**-6	7.46	543.6681	3009	4.04

TABLE 7

Computational Results for Dembo Problem No. 8,

Using Coarse Initial Constraint Tolerance and Final Tolerance of 10^{-6}

Prob.	FCN Calls	Grad Calls	Equiv. FCN C.	Newton Avg.	FCN Calls	Grad Calls	Equiv. FCN C.	Newton Avg.
	INITIAL TOL=10**-6				INITIAL TOL=10**-4			
8A	86[illegible]	52	1233	3.18	[illegible]	5[illegible]	1030	1.74
8B	[illegible]4	60	1414	3.58	6[illegible]	[illegible]	1053	[illegible]
8C	136[illegible]	7[illegible]	19[illegible]	4.99	[illegible]	[illegible]	1101	[illegible]
Total	3[illegible]	[illegible]1	45[illegible]	3.95	2030	1[illegible]4	318[illegible]	[illegible]

Total Execution Time, Print Level 1

1[illegible].[illegible] seconds (INITIAL TOL=10**-6)

[illegible].5[illegible] seconds (INITIAL TOL=10**-4)

The above runs include all improvements described.

TABLE 8

Computational Results* for Rijckaert and Martens GP Problems

Prob No.	WATFIV Time	Std. Time	Their S.T.	Their Opt.	Our Opt.	FCN Calls (n_f)	Grad Calls (n_g)	Equiv. FCN C. (n_e)	Newton Avg.	Reason for Term.	Notes
1	0.52	.030	.003	.01208	.01210	388	37	536	1.52	K.T.	**
2	0.14	.008	.002	6300	6299.7	64	9	91	1.39	F.C.	
3	0.48	.028	.004	126344	126306	194	25	294	0.0	F.C.	
4	0.48	.026	.039	3.1681	3.1442	118	9	172	1.40	F.C.	
5	0.24	.014	.003	623015	623277	24	14	140	0.16	F.C.	
6	1.36	.080	.017	29.5985	29.2282	207	20	367	1.26	F.C.	
7	1.36	.080	.028	29.5985	29.2256	245	20	405	1.62	F.C.	
8	0.93	.055	.221	178.478	181.370	253	21	400	3.15	F.C.	
9	0.11	.006	.001	11.91	11.9002	29	5	39	0.70	K.T.	
10	0.18	.010	.001	-83.21	-83.26	124	13	163	1.74	F.C.	
11	0.25	.014	.003	-5.7398	-5.7398	109	13	161	1.08	F.C.	
12	0.89	.052	.013	-6.0482	-6.0483	219	25	419	1.26	F.C.	
13	1.26	.075	.026	7049.24	7082.93	361	28	577	2.08	F.C.	
14	6.11	.363	.024	1.1436	1.1436	1819	112	2939	3.15	F.C.	
15	1.10	.065	.018	0.2015	0.20566	272	18	452	2.37	K.T.	
16	1.16	.069	.019	.1966	.1966	302	19	492	2.53	K.T.	
17	4.03	.239	.022	.1406	.1406	939	46	1445	1.99	F.C.	**
18	F	F	.034	1.81818		Could not find feasible point					
19	1.51	.089	.067	17486.	17485.9	311	27	527	1.94	K.T.	
21	1.62	.096	.094	-1237.55	-1252.8	260	33	590	1.04	F.C.	
22	1.68	.100	.175	-375.784	-379.96	241	27	484	2.03	F.C.	
24	1.05	.062	.175	97.591	97.569	140	16	320	0.75	K.T.	

Problem 23 was the same as Dembo #2 so it was not rerun; Problem 20 had an unresolved typographical error.

*The feasibility tolerance used was 10^{-4} in contrast to the stricter tolerance of 10^{-5} used by Rijckaert and Martens. This difference would tend to bias results in favor of GRG.

**Tolerance controlling termination had to be tightened by a power of 10.

Note that GRG is competitive or superior in it's standard time on the larger problems, 19 thru 24. Since all times except two are on the order of 1 second, the printing of some output by GRG (which may consume a large fraction of run time in these cases) and the previously mentioned difficulties with using standard times, imply that these comparisons must be taken with a large grain of salt.

An enhancement of the GRG code, described in [5], uses quadratic extrapolation to compute initial estimates of basic variables prior to solution of the nonlinear constraint equations in contrast to tangent vector extrapolation [4] used in the runs described above. Some of the Dembo and Rijckaert-Martens problems were used in tests to compare the two extrapolation schemes. The results, displayed in Tables 9 and 10, (which exhibit minor discrepancies with the results in Tables 5-8 owing to minor differences in tolerances and strategies used) show the superiority of quadratic extrapolation for these problems.

Conclusions

Conclusions to be drawn from these experiments are:

1. "Standard time," as defined by Colville in [1], is an inadequate means of compensating for different computing environments when attempting to compare optimization algorithms. Improved procedures are needed.
2. GRG, representing the class of general purpose NLP algorithms, competes well with special purpose geometric programming codes in solving geometric programs.

TABLE 9

Performance of GRG Using Tangent Vector Extrapolation

Test Problem No.	Function Calls (n_f)	Gradient Calls (n_g)	Equiv. Function Calls (n_e)	Newton Calls (NC)	Newton Failures (NF)	Newton Iterations (NI)	Newton Average NI/(NC-NF)	Execution Time (Sec.)	Standard Time
D2	17	6	47	7	0	8	1.14	0.18	0.0107
D4A	141	18	285	56	1	68	1.24	0.76	0.0452
D5	165	14	277	50	3	114	2.43	0.75	0.0446
D8A	1398	72	1902	308	60	1057	4.26	5.17	0.3072
R6	231	19	383	75	4	139	1.96	1.48	0.0879
R8	253	21	400	59	6	186	3.51	3.15	0.1872
R12	219	25	419	93	4	117	1.31	1.01	0.0600
R14	670	45	1120	182	77	465	2.66	3.21	0.1907
R15	272	18	452	78	4	185	2.50	1.21	0.0719
R16	302	19	492	83	5	210	2.69	1.27	0.755
Total	3668	257	5777	991	94	2549	2.84	18.19	1.0808

TABLE 10

Performance of GRG Using Quadratic Extrapolation

Test Problem No.	Function Calls (n_f)	Gradient Calls (n_g)	Equiv. Function Calls (n_e)	Newton Calls (NC)	Newton Failures (NF)	Newton Iterations (NI)	Newton Average NI/(NC-NF)	Execution Time (Sec.)	Standard Time
D2	17	6	47	7	0	8	1.14	0.18	0.0107
D4A	124	18	268	56	1	51	0.91	0.75	0.0446
D5	145	14	257	50	3	94	2.00	0.74	0.0440
D8A	1054	62	1488	278	43	743	3.16	4.20	0.2496
R6	217	19	369	79	5	121	1.64	1.46	0.0868
R8	198	19	331	58	6	132	2.54	0.89	0.0529
R12	170	22	346	82	1	79	0.98	0.89	0.0529
R14	452	37	822	138		291	2.29	2.47	0.1467
R15	253	18	433	81	5	163	2.14	1.19	0.0707
R16	251	20	451	85	2	157	1.89	1.29	0.0766
TOTAL	2881	235	3500	914	77	1839	2.20	14.06	0.8354
% Reduction in Total from Tangent Vector Extrapolation	21.5	8.6	39.4	7.8	25.7	27.8	21.5	22.7	22.7

3. Certain modifications in solution strategy can strongly affect the performance of GRG. Among these are: when the approximate hessian is reset, the logic used in basis inversion to decide when a variable at bound is to enter the basis, and the order of extrapolation (linear or quadratic) used to obtain initial estimates of the basic variables.
4. Certain parameter settings strongly affect GRG performance: in particular, the tolerance used to determine which constraints are binding, and the tolerance used to terminate the algorithm.

In closing, we note some things left undone but worth doing. GRG could easily be made more convenient and efficient on geometric programs by coding a special subroutine to compute first partial derivatives. This would use the fact, that if the ith term in the program is

$$t_i = c_i \prod_{j=1}^{n} x_j^{a_{ij}}$$

then

$$\frac{\partial t_i}{\partial x_k} = \frac{a_{ik} t_i}{x_k}$$

Hence, if the terms are stored when computing the constraint and objective values, their partial derivatives are available with little additional effort. This would reduce the time required to compute the gradient of a function from the time required in these runs (nt_f, where t_f is the time required to evaluate the function and n is the number of variables) to little more than t_f. Special input subroutines could be coded to enable the user to specify the problem by inputting only (a) the constants c_i, (b) the exponent matrix a_{ij} and (c) which terms appear in which problem functions. Currently, all problem functions must be coded directly. These enhancements would transform GRG into a "special purpose" geometric programming code.

Some additional experiments appear useful. Geometric programs can be transformed into exponential form by the change of variables

$$x_j = e^{y_j}$$

which transforms the ith term into

$$t_i = c_i \exp(\sum_j a_{ij} y_j)$$

Evaluation of t_i then requires only one transcendental computation rather than one for each fractional a_{ij}. In addition, y_j is a free variable (if x_j has no upper bound), and the problem functions become convex if all c_i are positive. Some problems should be solved using both forms, to see which yields smallest solution times. In addition, tests of GRG and some good geometric programming codes should be run on the same computer, in order to remove the factor of standard times from obscuring the comparisons.

REFERENCES

[1] Colville, A.R., "A Comparative Study of Nonlinear Programming Codes," IBM New York Scientific Center Report 320-2949, 1968.

[2] Dembo, R., "A Set of Geometric Programming Test Problems and Their Solutions," Working Paper No. 87, Department of Management Sciences, University of Waterloo, Waterloo, Ontario, December 1974.

[3] Jain, A., "The Solution of Nonlinear Programs Using the Generalized Reduced Gradient Method," Technical Report SOL 76-6, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California, March 1976.

[4] Lasdon, L.S., A. Waren, A. Jain, and M.W. Ratner, "Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming," Technical Report SOL 76-3, Systems Optimization Laboratory, Department of Operations Research, Stanford Univ., Stanford, Ca., February 1976.

[5] Lasdon, L.S., A. Waren, A. Jain, and M.W. Ratner, "GRG System Documentation," Tech. Memo. CIS-75-01, Computer and Information Science Department, Cleveland State University, Cleveland, Ohio, November 1975.

[6] Rijckaert, M.J., and X.M. Martens, "A Comparison of Generalized Geometric Programming Algorithms," Report CE-RM-7503, 1975, Katholieke Universiteit Leuven, Belgium.

APPENDIX A

Starting Values of Variables Used for the Rijckaert-Martens Problems in GRG Runs

Problem No.	
1	41.0, 140.0, 4.1, 2.1
2	54.0, 126.0, 102.2
3	375.0, 0.17, 0.73, 5.1
4	1.25, 3.75, 3.8, 1.8, 3.9, 1.95, 2.14, 4.2, 0.85, 3.0, 3.3
5	21.5, 67.0, 33.0, 1.6
6	0.5, 0.3, 0.56, 1.1, 0.5, 1.05, 0.56, 1.5
7	0.5, 0.3, 0.56, 1.1, 0.5, 1.05, 0.56, 1.5
8	0.67, 1.5, 0.44, 1.38, 1.57, 0.6, 0.77
9	0.41, 660.0
10	44.1, 11.0, 0.65
11	4.66, 1.23, 0.28, 2.82
12	3.23, 1.32, 0.51, 8.96, 1.11, 0.9, 0.2, 8.3
13	290.0, 2040.0, 2550.0, 273.0, 148.0, 327.0, 143.0, 594.0
14	1.05, 18.15, 3.95, 0.69, 0.18, 0.69, 0.22, 2.46, 0.60, 0.05
15	0.36, 1.08, 0.36, 0.39, 0.09, 0.16, 0.1, 0.21, 0.05, 0.45
16	0.36, 1.08, 0.36, 0.39, 0.09, 0.18, 0.1, 0.21, 0.05, 0.45
17	3.5, 11.4, 3.[illegible], 0.02, 0.4, 1.5, 0.19, 0.55, 0.18, 3.0, 0.23
18	0.2, 0.21, 0.1, 0.96, 0.3, 0.5, 0.003, 0.04, 0.2[illegible], 2.8, 1.2, 0.24, 0.17
19	2600.0, 9.9, 60000.0, 1000.0, 44000.0, 274.0, 0.06, 45.0
21	900.0, 9000.0, 48.0, 4500.0, 1000.0, [illegible], 92.0, [illegible], 1.8, 150.0
22	5.9, 0.5, 0.68, 6.1, [illegible], 0.2, [illegible], 0.36, 0.26
24	0.4, 1.0, 0.9, 0.05, 0.38, 0.11, 1000.0, 37.0, 750.0, 0.2

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL-76-1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SOLVING GEOMETRIC PROGRAMS USING GRG-RESULTS AND COMPARISONS.		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) M. RATNER, L.S. LASDON A. JAIN		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0267 N00014-75-C-0865 N00014-75-C-0240 (Case)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research Stanford University Stanford, California 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-064 NR-047-143 NR-047-105
11. CONTROLLING OFFICE NAME AND ADDRESS Operations Research Program Code 434 Office of Naval Research Arlington, Virginia 22217		12. REPORT DATE May 1976
		13. NUMBER OF PAGES 19
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

14 6 9 10 15 16 11 12 22p.

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Geometric Programs, GRG, Optimization Codes, Standard Time.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE ATTACHED

DD FORM 1473 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

408765 LB

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract. SOL 76-1

This paper describes the performance of a general purpose GRG code for nonlinear programming in solving geometric programs. The main conclusions drawn from the experiments reported are:

(1) GRG competes well with special purpose geometric programming codes in solving geometric programs and,

(2) "Standard Time," as defined by Colville, is an inadequate means of compensating for different computing environments while comparing optimization algorithms.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)